Tools for Pattern-Based Transformation of OWL Ontologies

Ondřej Šváb-Zamazal[1], Enrico Daga[2], Marek Dudáš[1], and Vojtěch Svátek[1]

[1] Department of Information and Knowledge Engineering,
University of Economics, W. Churchill Sq.4, 130 67 Prague 3, Czech Republic
{svatek|ondrej.zamazal}@vse.cz
[2] STLab, ISTC-CNR, Via Nomentana 56, 00161 Rome, Italy
enrico.daga@cnr.it

1 Motivation for Pattern-Based Ontology Transformation

The high expressivity of the OWL ontology language often allows to express the same conceptualisation in different ways. A simple example is the difference between 'class-centric' and 'property-centric' modelling style, such that the same notion is modelled as a class in the former (e.g. 'Purchase') and an object property in the latter (e.g. 'bought_from'). Such heterogeneity is an obstacle to reusing ontologies in advanced semantic web scenarios. In particular, two ontologies modelled in different styles are difficult to *match* or to *import* to one another, as few matching systems support complex matching structures that bridge style heterogeneity, never mind considering schema merging and/or data migration. Furthermore, opting for a style when designing an ontology may have impact on the usability and performance of *reasoners*, as some features cause performance problems for certain reasoners. Semi-automatic transformation of the modelling style of existing ontologies, with the help of tools to be presented in the demo, will alleviate such problems.

The whole tool suite consists of the *PatOMat* Transformation Framework (computational core of the approach), the *XDtools* ontological engineering framework with its *Transformation Wizard* for ontology adaptation (wrt. a content pattern to be imported), and the graphical *Transformation Pattern Editor*.

2 *PatOMat* Transformation Framework

The central notion in the *PatOMat* framework[3] is that of *transformation pattern* (TP). A TP contains two *ontology patterns* (the source OP and the target OP) and the description of transformation betweem them, called pattern transformation (PT). The representation of OPs is based on the OWL 2 DL profile, except that *placeholders* are allowed in addition to concrete OWL entities. An OP consists of *entity declarations* (of placeholders and/or concrete entities), *axioms* and *naming detection patterns*; the last capture the naming aspect of the

[3] [1] provides more details about the (earlier version of the) framework, and at http://owl.vse.cz:8080/tutorial/ there is a fully-fledged tutorial for the current version.

OP, which is important for its detection. A PT consists of a set of *transformation links* and a set of *naming transformation patterns*. Transformation links are either *logical equivalence relationships* or *extralogical relationships* (holding between two entities of different type). Naming transformation patterns serve for generating names for target entities. Naming patterns range from *passive naming operations*, such as detection of a head noun for a noun phrase, to *active naming operations*, such as derivation of verb form of a noun.

The framework prototype implementation is available either as a *Java library* or as three *core services*.[4] The Java library is directly used in the XD Transformation Wizard, see Section 3. The whole transformation is divided into three steps that correspond to the three core services:

- *OntologyPatternDetection* service takes the TP and an ontology on input, and returns the binding of entity placeholders on output, in XML. The naming detection patterns of the source OP are first processed. As a result of applying the naming aspect, bound placeholders arise that are placed to the FILTER component of a SPARQL query (generated according to axioms in the source OP) before its execution.
- *InstructionGenerator* service takes the binding of placeholders and the TP on input, and returns transformation instructions on output.
- *OntologyTransformation* service takes transformation instructions and the original ontology on input, and returns the transformed ontology on output.

The third service is partly based on OPPL [2] and partly on our specific implementation over OWL-API.[5] In contrast to plain OPPL, we use naming constraints and we *decompose* the process of transformation into parts, which enables user intervention within the whole workflow.

3 Support Tools for Ontology Transformation

A specific case of ontology import is the import of *ontology content patterns* [5] (as small 'best-practice' chunks of ontological knowledge) into legacy ontologies, which, in turn, often have to undergo transformation. For example, when an ontology is to be adapted to the *AgentRole* pattern,[6] often seemingly 'natural' classes have to be changed to 'role' classes. To make such operations rapid and smooth, we decided to closely integrate *PatOMat*, through a *Transformation Wizard*, with *XDtools*,[7] an ontological engineering environment specifically tailored for ontology content patterns (CPs) exploitation.

The *eXtreme Design* (XD) approach [3] introduced a new generation of methods for ontology design. Instead of performing language-oriented operations, e.g. instantiate a class, define a subclass, etc., the designer handles larger chunks of

[4] All accessible via the web interface at `http://owl.vse.cz:8080/`.
[5] `http://owlapi.sourceforge.net/`
[6] `http://ontologydesignpatterns.org/wiki/Submissions:AgentRole`
[7] `http://extremedesign.sourceforge.net`

ontologies (patterns), which leads to modularity and compliance with good design practices. Drastic decrease of certain frequent mistakes was observed when the *XD methodology* [4], the *XDtools application*, and the *OntologyDesignPatterns.org* community-based pattern catalogue were jointly used. The tool provides an Eclipse perspective that includes, among other, the *XD Specialization Wizard* that guides the user in the operation of CP specialization [5].

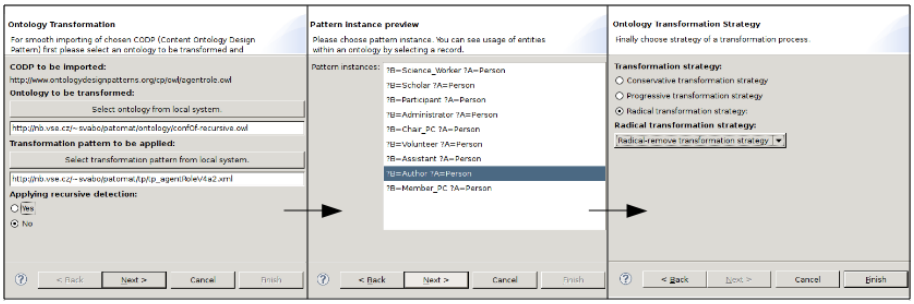

Fig. 1. Step-by-step ontology transformation using XD Transformation Wizard

The *XD Transformation Wizard* can be invoked (as Transform/Import) while right-clicking on an ontology from the Eclipse project or from the 'ODP Registry' view. This action chooses the *content pattern* to be imported into an ontology. On the first page of the wizard (see Figure 1) an user selects an ontology from a local system or from the web. The suitable *transformation pattern* is also selected. Finally, the user can turn on *recursive detection* of pattern occurrences (over a taxonomy), as supported by a reasoner. After clicking on the 'Next' button, the import and detection operations are performed. The second page allows the user to select from *pattern instances* (placeholder bindings) returned by the detection phase. By clicking on an entity, the user can also display its usage within the ontology, for better overview. The final page of the wizard offers the selection of finer *transformation strategy*. After the transformation, the modified ontology is stored either as a new ontology or as an ontology version.

Finally, in order to support the *authoring and update* of transformation patterns, we developed a *Transformation Pattern Editor* (TPE). It allows graphical modeling (see Figure 2) and export/import from/to the (XML-based) transformation pattern notation. The upper-left and upper-right pane contain the source and target OPs, while the bottom one contains the PT. All elements are displayed using the 'auto-layout' function. TPE is available as a plugin for Eclipse and uses the Graphical Editing Framework.[8]

Additional information and installation instructions about the XD Transformation Wizard and the TPE are at `http://owl.vse.cz:8080/tools.html`.

[8] `http://www.eclipse.org/gef/`

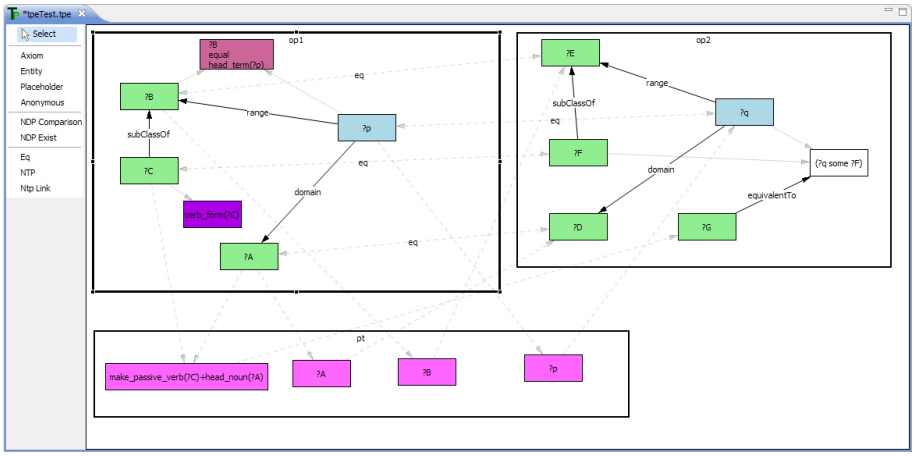

Fig. 2. Transformation Pattern Editor in action

The demo will feature several variants of wizard-based transformation wrt. two best-practice content patterns (for role-based modelling and for reified participation). About ten other TPs can be invoked over RESTful services. TPE can be shown in the design of a new TP or modification of an existing one.

4 Future Work

The imminent development plan for the XD Transformation Wizard is its integration into the NeOn toolkit. As enhancement for TPE, online selection of entities from catalogued content patterns is envisaged. There is also space for improvement in the automatic layout of entities. Finally, we obviously plan to integrate TPE with XDtools, and also to link the transformation patterns repository to the 'XD Repository' of XDtools.

The research has been partially supported by the CSF grant no. P202/10/1825.

References

1. Šváb-Zamazal O., Svátek V., Iannone L.: Pattern-Based Ontology Transformation Service Exploiting OPPL and OWL-API. In: EKAW-2010, Lisbon, Portugal, 2010.
2. Egaña M., Stevens R., Antezana E.: Transforming the Axiomisation of Ontologies: The Ontology Pre-Processor Language. In: OWLED 2008.
3. Presutti V., Daga E. Gangemi A., Blomqvist E.: eXtreme Design with Content Ontology Design Patterns In: Workshop on Ontology Patterns at ISWC 2009.
4. Blomqvist E., Presutti V., Daga E. Gangemi A.: Experimenting with eXtreme Design. In: EKAW-2010, Lisbon, Portugal, 2010.
5. Presutti V., Gangemi A.: Content ontology design patterns as practical building blocks for web ontologies.: In Proceedings of ER2008. Barcelona, Spain, 2008.